A Fully Bypassed Six-Issue Integer Datapath and Register File on the Itanium-2 Microprocessor

Eric S. Fetzer, Mark Gibson, Anthony Klein, Naomi Calick, Chengyu Zhu, Eric Busta, and Baker Mohammad

***Abstract*—The six-issue integer datapath of the second-generation Itanium Microprocessor is described. Pulse techniques enable a high-speed, 20-ported, 128-entry, 65-bit register file with only 12 wordlines per register. A four-stage operand bypass network achieves a fully bypassed design with operands sourced from 34 locations with 16 destinations. To control this network, over 280 bypass comparators are utilized. Using half a clock for execution and half a clock for bypass, each result is available for the next instruction. Functional units are pre-enabled, reducing power consumption by 15% while eliminating a stage of result muxing and improving performance. The part is fabricated in a six-layer, 18-μm process and operates at 1.0 GHz at 1.5 V, consuming less than 130 W in about 420 mm^2.**

***Index Terms*—Digital integrated circuits, integrated circuit design, integrated circuit noise, microprocessors, registers.**

I. Introduction

THE Itanium-2 microprocessor, the second implementation of the Itanium architecture, features an explicitly parallel architecture. This architecture lends itself to a highly superscalar implementation. Highly superscalar designs require larger register files (RFs) to feed multiple execution units and complex bypass networks to keep data freely moving through the system. The Itanium-2 microprocessor incorporates a six-issue integer datapath (IEU) with a 20-ported, 128-entry, 65-bit-wide RFs. To prevent data hazards, integer operands are fully bypassed through four stages of bypass multiplexing with each of the 12 integer operands and four data cache addresses sourced from 34 possible results (RF, instruction field, two L1 data caches, six arithmetic and logic (ALU) EXE stages (Fig. 1), eight DET stages, eight WRB-stage integers, six WRB-stage multimedia, and various architected registers). All IEU operations require a half cycle for execution and a half cycle for bypass, allowing each ALU result to be used in the next cycle as a source for ALU or data cache addresses. In this paper, we detail circuit and analysis techniques used to complete this high-performance design.

Manuscript received March 15, 2002; revised June 10, 2002.

E. S. Fetzer, M. Gibson, A. Klein, and E. Busta are with Hewlett-Packard Company, Fort Collins, CO 80528-9599 USA (e-mail: eric.fetzer@hp.com; mark.gibson@hp.com; anthony.klein@hp.com; eric.busta@hp.com).

N. Calick and C. Zhu are with Intel Corporation, Fort Collins, CO 80525 USA (e-mail: naomi.b.calick@intel.com; chengyu.zhu@intel.com).

B. Mohammad is with Intel Corporation, Chandler, AZ 85226 USA (e-mail: baker.mohammad@intel.com).

Digital Object Identifier 10.1109/JSSC.2002.803948

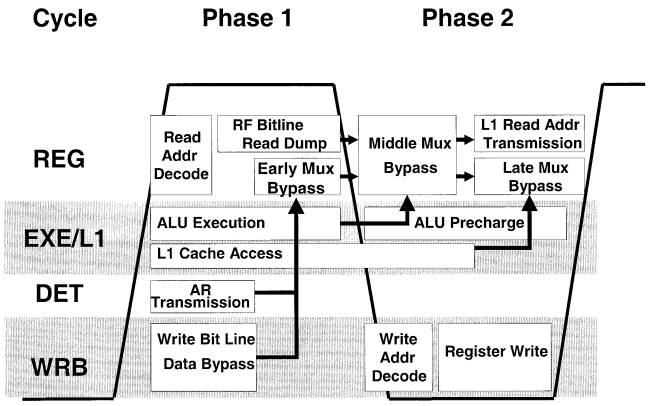


Fig. 1. IEU state and timing diagram.

II. Datapath

A. RF

The 20-ported RF is 2.2 mm^2 and incorporates 12 read and eight write ports. To accomplish all required bypassing, RF reads occur simultaneously with the first two stages of bypass and ALU execution (Fig. 1). As the RF read is dumped onto the read data bit line, the RF word line decoder is sent the address for the write. During the read, the write data is also bypassed along the write bit lines reusing these bit lines for a bypass from the six-issue multimedia unit to the integer datapath. Single-ended read performance is maintained with a two-level bit line structure saving wires and reducing delay (by removing self-timed path margin) relative to sense-amp designs of the past [1], [2]. Registers are banked in arrays of 16 with bit lines in the second metal layer. The bit lines are then repeated and promoted to the fourth metal layer to drive the outputs to the middle bypass mux in the integer and multimedia units (MMU).

In a standard register file, 20 ports would require 20 word lines per register. Such a design would have exceeded area and timing constraints. Double pumping the word lines achieves same cycle reads and writes with a single wire. Each register has only 12 word lines and a shared control wire (WRITEH). WRITEH determines the read/write directionality of the word line with selection internal to each register (Fig. 2). The word line decoder decodes read and write register addresses. To do this the decoder is clocked with a pulse on the rising and falling edge of the clock. The register address is statically decoded with the results of the high- and low-order bits being strobed by the double pumped pulse clock (PCK2X) generated by the circuit shown in Fig. 3. Pulses do not propagate well through many

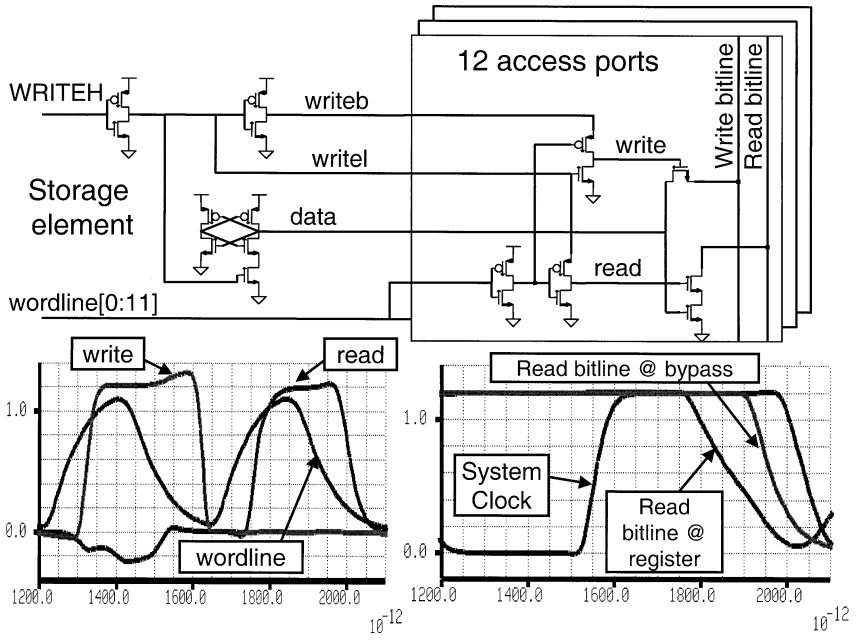


Fig. 2. Register file circuit and timing diagram.

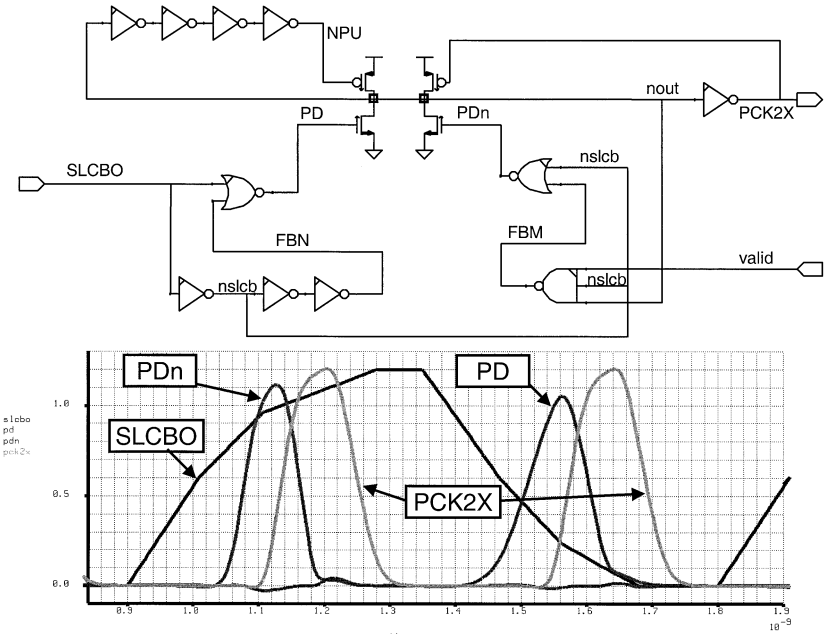


Fig. 3. Double-pumped pulse clock generator circuit and timing diagram.

stages of logic. To prevent pulse degradation, a pulsewidth-control feedback delay is inserted between the decoder and the word line. The word line muxing, internal to the register, captures pulses during the write phase of the system clock and holds the write signal at high value until the end of the phase, giving the write mechanism more than a pulse width to write data into the register. Since writes are single ended through a nFET pass gate, one leg of the cell is floated using a virtual ground, which improves timing and cell writeability. This technique is demonstrated in silicon to work at < 1 V.

B. Operand Bypass Datapath

The integer datapath bypassing is divided into four stages, to afford more timing critical inputs the least possible logic delay to the consuming ALUs. Critical L1 cache return data must flow through only one level of muxing before arriving at the ALU inputs, while DET and WRB data, available from staging latches, have the longest logic path to the ALUs. This allows the bypassing of operands from 34 possible results to occur in a half clock cycle, enabling a single-cycle cache access and instruction execution.

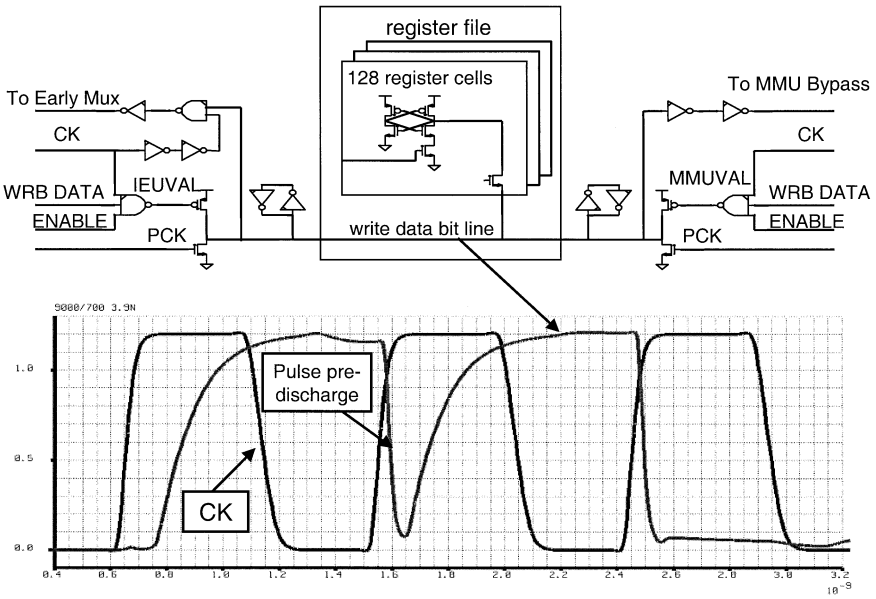


Fig. 4. WRB bypass circuit and timing diagram.

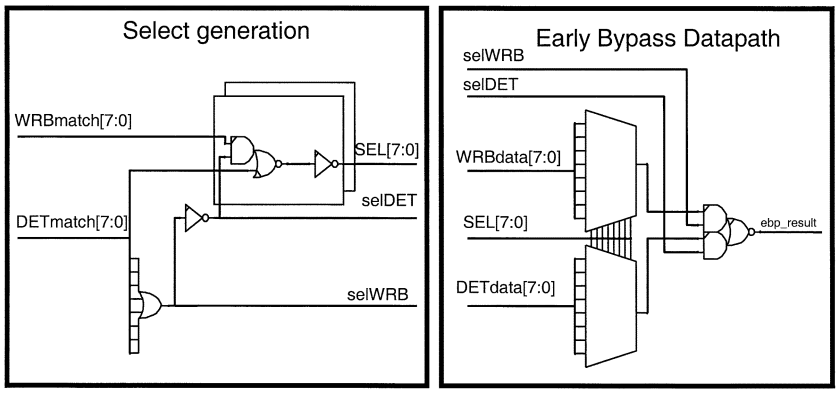


Fig. 5. Early bypass control and mux diagram.

The first stage of bypassing (Fig. 4) occurs on the RF write bit lines. The RF writes the value on the bit line during the low phase of clock (CK). With the MMU results on the right and the IEU results on the left, an enable signal determines which unit can assert a value onto the bus during the high phase of clock. The write bit line is pre-discharged to logical 0, electrical ground, on rising clock using a pulse clock (PCK) to discharge the bus via an nFET on each side of the RF. The bit line is pulled to logical 1 by the active unit during the high phase of clock. This provides a monotonic rising edge for the down stream muxes in bypass. To prevent high phase-dynamic logic in early bypass from receiving a glitch during the pulse predischarge, a delayed clock is ANDed with the write data bit line.

The eight merged WRB results, from the register file bypass and the eight integer DET results, driven by a staging latch, are muxed in the early bypass, with priority given to the DET values. Each clock cycle, 12 operands must be delivered to six ALUs, so there are 12 distinct early bypass muxes, two for each execution port. A standard dynamic implementation would require at least one mux select for each of the 16 mux inputs or 192 total select wires. In order to conserve wire tracks, an encoding scheme was used, where first the DET and WRB results are each combined in an 8:1 mux, using the same eight select lines. The selects are generated by domino logic implementing DET/WRB priority, which makes the selects valid for both the DET and WRB muxes. Then a 2:1 mux determines the priority between DET and WRB. This scheme uses only ten mux selects, saving 96 wires total (Fig. 5). The early bypass results are available at the end of the first phase of the clock cycle.

The six ALU EXE results and the register file read data are not valid until late in the first half of the clock cycle. So, these seven data elements and the early bypass result are combined in the third muxing stage, the middle bypass. The domino bypass mux operates in the second half of the clock. In order to alleviate the need for a latching element, bus holders and overlapping clocks were used to allow the high-phase-valid signals to transition to the low phase domino mux circuitry. On all six issue ports, the middle bypass output is forwarded to the late bypass mux, for final determination of the ALU inputs. In addition, four middle bypass results are sent directly to the L1

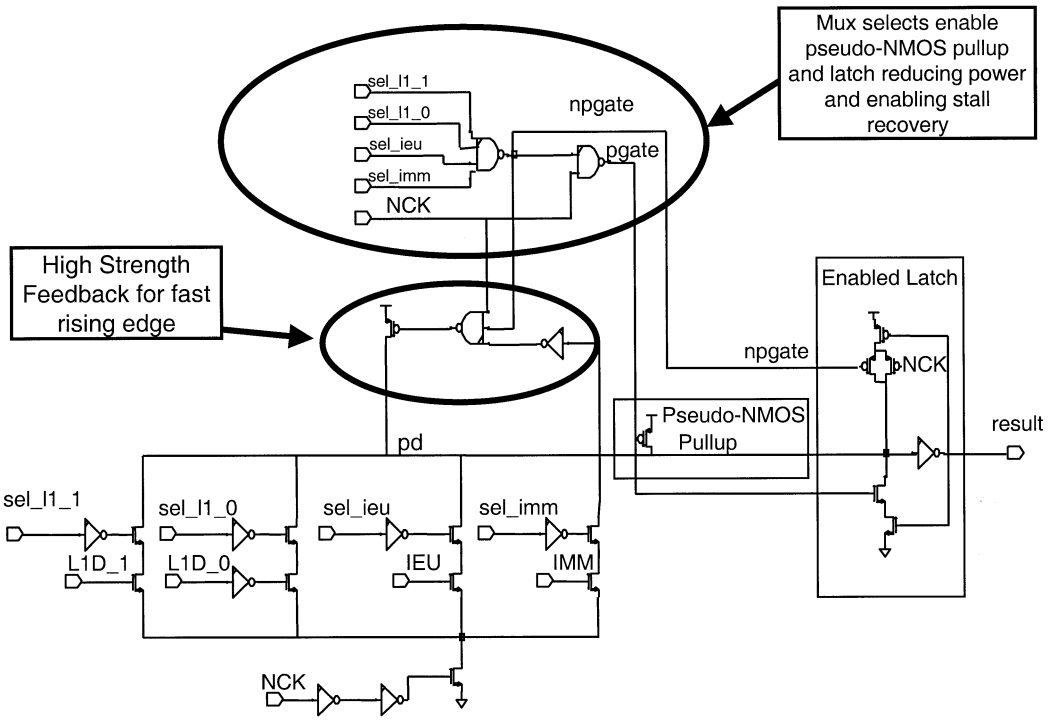


Fig. 6. Late bypass circuit diagram.

data cache to provide two load and two store addresses each clock cycle.

The final stage of the bypassing network, the late bypass, selects two L1 data cache returns, the middle bypass result and the immediate field encoded in the instruction. (Fig. 6) The L1 data cache uses single-ended reads from both sides of a single SRAM cell, so while the return data is electrically the same for both terms, one of the values is logically inverted. This polarity prevents the use of a domino circuit, so instead a clocked pseudo-nMOS approach is used. A clocked enable controls the pseudo-nMOS pull-up device; in combination with a delayed clock gating off the pull-down stack, this creates a storage element on the pseudo-nMOS node PD. The controlling clock (NCK) is inverted from CK, making the late bypass output valid and stable for the first half of the clock cycle and suitable for consumption by the integer ALUs.

The pull-down structure of the pseudo-nMOS mux is a four term AND-OR structure, with the four mutually exclusive mux selects on top of the n-fet pull-down stack. In order to better balance the rise and fall times (120 and 80 ps, respectively) of this circuit, a high-strength feedback circuit creates a fast rising edge on the pseudo-nMOS node when the pull-down network is disabled. If the PD node is logical 1 (VDD), the high strength pull-up device is gated off so the nFET term only has to overcome a much weaker pull-up device. This increases the speed of this circuit and reduces the transient drive-fight current seen in traditional pseudo-nMOS circuitry. The latch enable is the AND of the four active-low select terms and NCK. The clock gating off the pull-down stack is delayed, which ensures that the PD node will hold its value as the latch is closing. While traditional pseudo-nMOS circuits often consume a great deal of power, this gate only draws current during a small portion of the clock cycle, around 20%. The late bypass mux output is sent directly to the ALUs. On two of the six execution ports the late bypass result is used to drive store data to the L1 cache.

C. Functional Units

The Itanium-2 IEU contains multiple functional units in each integer execution pipeline. The output of the late bypass delivers operands to each of the functional units in a given pipeline. The functional units in the IEU convert the static result from the late bypass muxes to a dynamic signal using a pulse clock evaluate dynamic entry latch. The explicit pulse clock allows a significant amount of logic to be implemented in the entry latch with only a modest increase in the hold time requirements over the typical entry latch [3] used on The Itanium-2 microprocessor. The ALU inputs incorporate a variable 5-bit shift into the dual rail entry latch before each adder (Fig. 7). The pulse evaluate requires careful attention to evaluate margin and charge sharing after the pulse.

The functional units in the IEU are pre-enabled so the functional unit does not receive a clock signal unless the result will be valid. The output circuit of any unused unit will remain in precharge, which disables the local pulldown. Only one functional unit will pull the shared result bus down during a given clock cycle, guaranteeing logically correct operation. This eliminates the need for a result mux to merge the results of the multiple functional units, improving overall delay by about 5%–7%. It also results in a dramatic power reduction of nearly 15%. Finally, the result busses utilize active noise cancellation [3] circuits to reduce the effects of noise events that occur on the long routes.

D. IA-32 Support

The IEU supports full IA-32 functionality through four key circuits: ALU, PRODF, MISC, and LAGEN. The ALUs perform all the IA-32 integer operations, including adds and logical

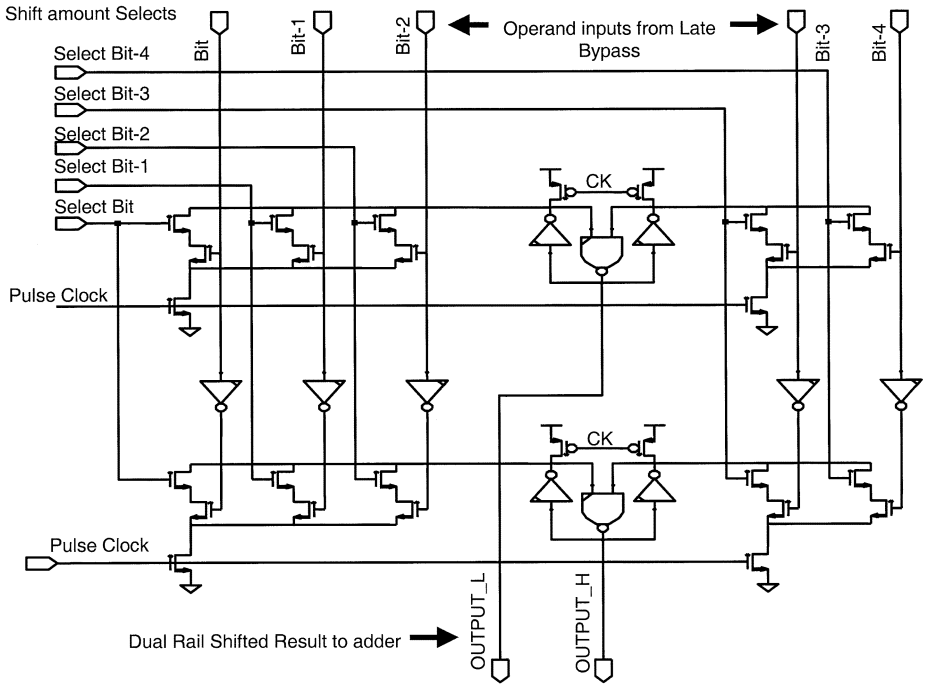


Fig. 7. ALU dynamic entry latch with dual rail outputs.

operations with its existing 64-bit ALU datapath (all the ALU datapath blocks have 8-, 16-, and 32-bit controls for IA-32 operations). The PRODF block generates all the flags that are needed for IA-32 while the MISC block takes care of IA-32 carry bit generation for IA32 adds. LAGEN, the IA-32 Linear Address Generator, generates memory addresses for IA-32 instructions. The LAGEN block generates the effective address, which is the sum of base register and displacement and the linear address, which is the sum of effective address and segment base. With only 1/2 a cycle, the LAGEN needs to do two 32-bit adds and two mux operations in less than 500 ps. To accomplish this, a 3:2 carry save adder (CSA) is followed by a 32-bit Ling [4] adder.

Dual-rail dynamic logic is employed to build the entire LAGEN. The 32- bit adder is leveraged heavily from the 64-bit adder used in the ALU. The carry-look-ahead adder takes advantage of the simplified 4-bit carry Ling term [4], [5] to build a very well-balanced adder with no nFET stack of more than three pulldown FETs. The precharge clocks of the critical path are carefully planned to enable the maximum number of consecutive unclocked pulldown domino stages connected in serial for maximum speed. In Fig. 8, the cells labeled D1 require clocked pulldown stages. The dynamic circuits, labeled D2, do not require clocked nFETs in the pulldown stack because the precharge timing of upstream gates prevents discharge during their precharge phase.

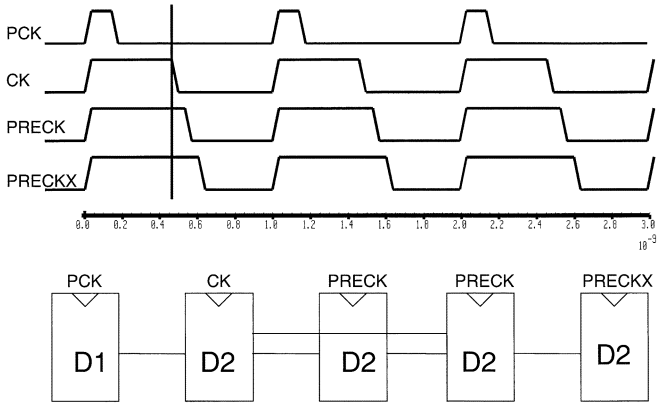


Fig. 8. Clock relationships used to reduce precharge drive-fights in unclocked dynamic pull-down networks.

III. Bypass Control

The bypass datapath is capable of delivering two operands to functional units of each of the six integer pipelines plus four operands for L1 cache addresses. Operands come from the integer register file by default, however, staged results in the EXE, DET, or WRB for the same register must take precedence over register file data as they represent more recent values of the register. Bypass control is primarily responsible for determining where each of the 12 operands comes from, steering the bypass datapath appropriately and determining whether data hazards exist for any of the required operands.

A. Bypass Computation

In each cycle, for each of 12 operands, bypass control compares the seven bit operand register ID with the destination register IDs of: six EXE stage results, two EXE stage cache load returns, six DET stage results, two DET stage cache load returns, six WRB stage results, two WRB stage cache load returns. This requires 288 comparators. Operand ID matches with any of these 24 in-flight result IDs are qualified (to assure that both operand and result are valid), prioritized (more recent results take precedence over older results in the case of multiple matches), encoded and driven to the muxes in the bypass datapath. All this is done with custom dynamic logic in the first phase of the REG stage, in approximately 500 ps.

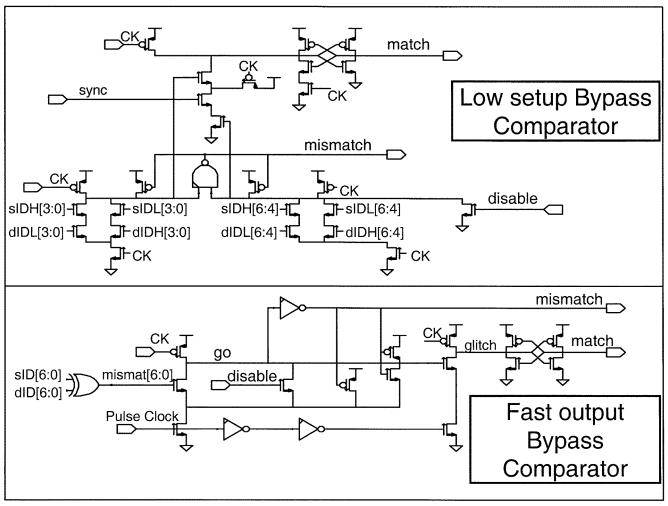


Fig. 9. Bypass control comparator circuits.

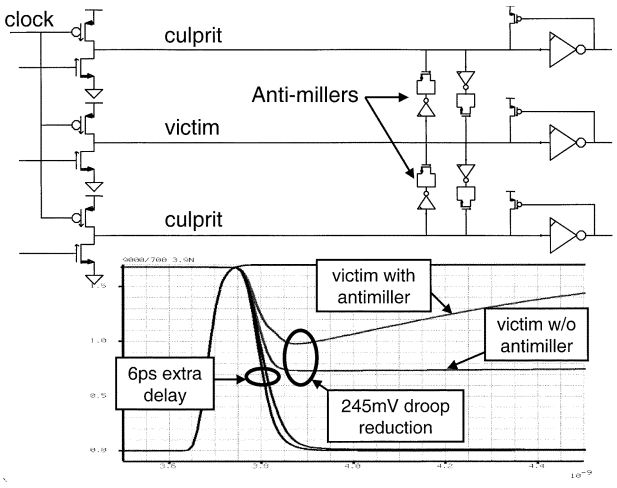


Fig. 10. Active noise cancellation circuit and waveform.

The 7-bit register ID comparators used produce both match and mismatch dynamic signals. Mismatch signals allow timely prioritization when result register IDs from multiple pipe stages match a given operand ID. For example, no WRB stage result can be bypassed if there is a more recent DET stage result register ID match. The eight DET stage mismatch signals are used to qualify any WRB stage match signal before generating the mux selects for a WRB stage bypass. An equivalent static implementation would not have met timing requirements.

B. 7-Bit Dynamic Comparators

Two versions of 7-bit dynamic comparators are used in bypass control. The first version of the comparator, shown in Fig. 9, receives a static copy of the operand register ID (sIDH[6:0] and sIDL[6:0]) and dual rail dynamic copies of the destination register IDs (dIDH[6:0] and dIDL[6:0]). The static operand ID bits stabilize just before the rising edge of the CK clock. The dynamic destination IDs arrive shortly after the rising edge of the CK clock.

The comparators look for any mismatch among the 7 ID bits and fire mismatch if any is found. A pulse (sync) generated locally by the arrival of the dynamic IDs, is simply buffered and driven out as the match signal, if no mismatch has been identified. Noise immunity is provided on the dynamic ID routes by interleaving the high and low dynamic ID bits with neighboring ID bits. This assures that at least one neighbor of each dynamic signal is stable for a given cycle. This comparator is used in the REG to EXE stage comparisons where the destination register IDs are not available until the very end of the REG stage.

The second version of the 7-bit comparator, shown at the bottom of Fig. 9, is used in the REG to DET and REG to WRB stage comparisons. In the DET and WRB stages the destination IDs are all kept locally in bypass control latches and can easily make setup to the rising edge of the CK clock. In this situation, the bitwise XOR of operand and result IDs is computed statically before the rising edge of CK. The XOR results (mismat[6:0]) are passed to a dynamic OR with a self-timed second stage to generate the logically opposite match and mismatch signals.

Shortly after the CK clock rises, the PCK pulse clock evaluates the intermediate (go) node. If any mismatch is asserted, the intermediate node disables the match signal and asserts the mismatch signal; otherwise, match asserts. Careful consideration of the evaluate speed of the first stage must be taken. The glitch node can droop significantly if the go node has not completely discharged when the pull down pulse has arrived. Note that both comparators have a disable signal to disqualify any match in the event of an invalid instruction or operand. The disable supersedes any comparison calculation by forcing the mismatch signal to assert.

IV. Analysis

A. Static Timing Analysis

With a design as large and complex as the Itanium-2 IEU, several unique timing challenges surfaced. As with most designs of this size, static timing analysis [6] was utilized extensively throughout the design. The goal of static timing analysis is to exhaustively model all possible paths within a block of circuitry and between circuit blocks. The pattern independence and runtime performance of static timing is achieved at some loss of accuracy compared to SPICE [7] modeling, which is pattern dependent and computationally expensive. Timing of circuits was performed in two phases: early timing and post layout timing.

Early timing, performed before layout parasitics are available, is a multi-pronged approach. First, extensive simulation of the timing of anticipated critical paths was performed using SPICE and estimated parasitics derived from floorplan and process information [8]. This allowed us to tune key architectural paths early in the design. Second, a simple gate delay accounting was done on the microarchitectural definition allowing us to redefine or reimplement features that could not be implemented while achieving our frequency goals. Finally, a schematic level static timing analysis was performed. Parasitics generated from the floorplan ensure that all routes greater than 200 μm had reasonable capacitances estimated.

Artwork timing, performed when layout exists to extract parasitic information, was phased in as mask data for different pieces of the unit became available. Our parasitic estimation

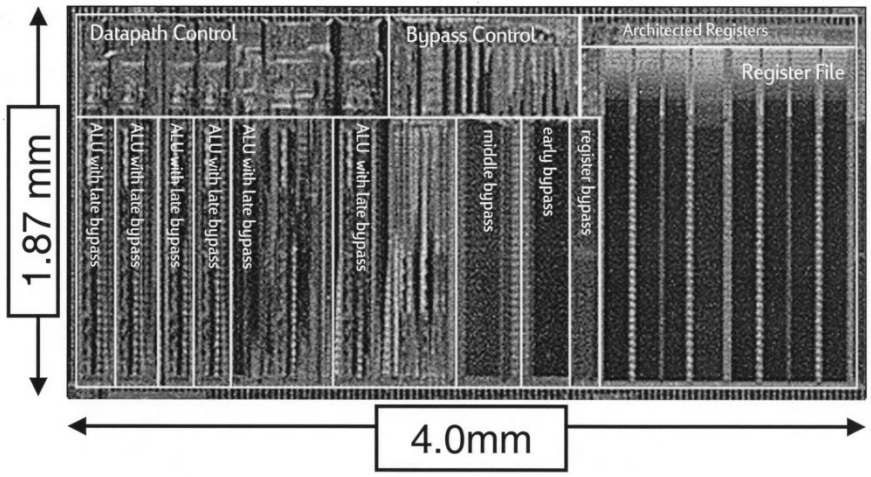


Fig. 11. Die micrograph.

flow, developed for early timing, was compatible with our incremental, hierarchical parasitic extraction of layout. Portions of the design that reached layout before others were converted to artwork timing while others used the early timing approach. This "mixed mode" simulation removed the normal dependencies between different parts of the IEU design, allowing timing verification of completed pieces of the design while others were still immature.

Two circuits in the design were especially difficult to manage: the register file and the late bypass mux. The register file featured wordlines with four potential transitions per clock cycle. The nature of static timing analysis assumes that each similar transition has a similar restraint. Since two different rising edges on the wordlines signals have two different timing restraints, it was impossible to model correctly. Instead, a hand-generated abstraction of the register file interface was developed from SPICE simulation. Similarly, the pseudo-nMOS nature of the late bypass mux could not be modeled accurately with static timing analysis. The static nature of the tool could not determine the effect of the feedback loop pullup on circuit performance, leaving the falling edge too slow and the rising edge too fast.

To simplifiy analysis and to achieve our skew goals clock signals were forced to have uniform, consistent edge rates and delays in this static timing methodology. Resistance and loading on clocks were not modeled in static timing analysis. Skew was a budgeted part of all timing paths. A separate clock extraction/analysis tool validated that clocks did not exceed the maximum skew. This allowed the timing model to converge before the final, low skew clocks were designed.

B. Noise

The IEU's aggressive circuit techniques and large dynamic precharge pulldown busses had the potential to be a severe problem, when combined with the greater than 2:1 aspect ratio of the aluminum interconnect. Traditional noise compensation approaches such as repeaters, shielding, spacing, or skewing receiver circuits excessively impact either the density of layout or speed of the signals too much to meet the design goals.

Many different methods for minimizing coupling with minimum impact to speed and area were implemented. All mutually exclusive mux select signals were routed together to provide a half shield for all bits. Static buses with timing margin are also routed together such that crosstalk induced glitches are cleared before the data are sampled in a latch. Interleaving phase-two dynamic signals with phase-one dynamic signals reduced crosstalk at the expense of precharge margin and evaluate performance but often proved a valuable tradeoff.

Even with these techniques there were cases where additional crosstalk suppression was necessary. Active noise cancellation circuits (anti-millers) [3], which dump opposite polarity charge on a victim wire when a culprit neighbor switches (Fig. 10), were used in cases where other crosstalk reduction techniques could not be used. These circuits are inserted between each adjacent bit in the datapath, reducing the crosstalk induced droop (or bounce) by 20%–30% on each edge with minimum impact on timing (only 5–10 ps).

With almost 2 million transistors and thousands of paths, simulation of each noise topology would be impossible. Instead the IEU utilized a two-step process developed to verify the design robustness. First, each cell was characterized in SPICE simulation to determine its input sensitivity to glitches. The propagation of the glitches is also characterized. Next, parasitic information, extracted from the layout, is used to calculate the coupling capacitance on every segment of the metal. Crosstalk-induced droop or bounce is then computed on each piece of interconnect. Several statistical techniques are used in the noise analysis to reduce pessimism, including relying on the probability that many unique neighbors are not likely to couple in unison. Crosstalk generated in one stage of logic is then propagated to the next stage using the characterization data from the SPICE runs on the receiver. If the crosstalk propagated through the receiver exceeds the defined parameters, an error is flagged. This method resulted in very few false errors and computation time was short.

V. Conclusion

The register file, bypassing and six ALUs fit in to a datapath that is 4.0 mm by 1.87 mm (Fig. 11). Power is highly data de-

pendent varying from 4 to 8 W, while averaging less than 6 W in real-world applications. At 1.5 V, the IEU operates at over 1.0 GHz with six ALUs, delivering four cache addresses and consuming two cache results each cycle. Careful static timing analysis delivered first silicon with no races and few unpredicted timing issues. Noise analysis was equally successful with no crosstalk-induced electrical failures detected on silicon.

REFERENCES

[1] R. Heald *et al.*, "Implementation of a 3rd-generation SPARC V9 64b microprocessor," in *ISSCC Dig. Tech. Papers*, Feb. 2000, pp. 412–413.

[2] M. Golden *et al.*, "A 500 Mhz, write bypassed, 88 entry, 90-bit register file," in *VLSI Symp. VLSI Circuits, Dig. Tech. Papers*, June 1999, pp. 105–108.

[3] S. Naffziger *et al.*, "Implementation of the next-generation 64b Itanium microprocessor," in *ISSCC Dig. Tech. Papers*, Feb. 2002, pp. 344–345.

[4] S. Naffziger, "A sub-nanosecond 0.5 μm 64b adder design," in *ISSCC Dig. Tech. Papers*, Feb. 1996, pp. 362–363.

[5] H. Ling, "High speed binary adder," *IBM J. Res. Dev.*, vol. 25, no. 3, p. 156, May 1981.

[6] R. B. Hitchcock Sr., "Timing verification and the timing analysis program," in *Proc. 1982 Design Automation Conf.*, pp. 594–604.

[7] J. H. Huang, "A robust physical and predictive model for deep-submicrometer MOS circuit simulation," in *Proc. IEEE Custom Integrated Circuits Conf. (CICC '93)*, May 1993, pp. 14.2.1–14.2.4.

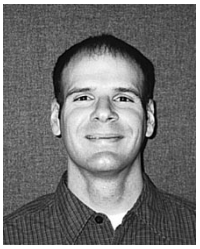
[8] N. Chang, V. Kanevsky, O. S. Nakagawa, K. Rahmat, and S.-Y. Oh, "Fast generation of statistically-based worst- case modeling of on-chip interconnect," in *Proc. IEEE ICCD*, Austin, TX, 1997, pp. 720–725.

Eric S. Fetzer received the B.S. degree in electrical and computer engineering from the University of Wisconsin, in 1996.

He is a Technical Contributor at Hewlett-Packard's Colorado VLSI Lab, Systems and VLSI Technology Division, where he developed methodologies and designed circuits for the Itanium-2 microprocessor. He currently holds two U.S. patents with several more pending. His interests include high-speed circuit design techniques, clocks and power distribution. He is presently working on the design of next-generation, high-performance IA-64 microprocessor.

Mark Gibson received the B.S. and M.Eng. degrees in electrical engineering from Cornell University, Ithaca, NY.

He first joined Digital Equipment Corporation to write microcode for large VAXen and to help design the CPU module for the first large Alpha processor-based system. He joined the CPU Design Laboratory, Hewlett-Packard Company, Fort Collins, CO, in 1994. Since then, he has performed verification, circuit design, and control design roles on both PA-RISC and IA-64 processor chips.



Anthony Klein received the B.S. degree in electrical and computer engineering from the University of Illinois, Urbana-Champaign in 1996.

He has been with Hewlett-Packard Company, Fort Collins, CO, since 1996. He is currently a Member of Technical Staff at the Microprocessor Technology Lab, Systems and VLSI Technology Division. Currently, he is working on the design of next-generation, high-performance IA-64 microprocessors. He has worked on the data cache of the PA-8500 microprocessor and aided in the design of the integer datapath for the Itanium-2 microprocessor.

Naomi Calick received the B.S. degree in electrical engineering from the University of California at Berkeley in 1993.

She has been with Intel Corporation, Fort Collins, CO, since 1993. Her experience with VLSI design began in Intel's Microprocessor Division 6, where she contributed to the design of the Pentium III processor. Later, she joined Intel's Enterprise Processor Division to work on the Intel Itanium architecture products, including Itanium 2 and its follow on.

Chengyu Zhu received the B.S. degree in electronics engineering from Fudan University, Shanghai, China, and the M.S. degree in electrical engineering from Purdue University, West Lafayette, IN.

He has been with Intel Corporation, Fort Collins, CO, since 1995, currently in the Enterprise Processors Division, where he is a Senior Design Engineer, involved in the physical design of numerous critical functional blocks of the integer execution unit and the level-three cache unit of the Itanium-2 processor. Prior to joining the current group, he was with the the Intel California Technology Development Division, engaged in the design and verification of the instruction decoder of the Pentium-II processor.

Eric Busta received the B.S. and M.S. degrees in electrical and computer engineering (with a computer science option) from the University of Wisconsin at Madison.

He is a Design Engineer at the VLSI Laboratory, Hewlett-Packard Company, Fort Collins, CO, where he is currently focusing on power reduction in high-performance microprocessors. Prior to joining the VLSI Laboratory, he spent several years working on high-reliability microprocessor systems for Collins Avionics.

Baker Mohammad received the B.S degree from the University of New Mexico, Albuquerque, and the M.S. degree from Arizona State University, Tempe, both in electrical engineering.

He first joined Intel Corporation, Chandler, AZ, in 1995, working on i960 RISC processor. He joined Intel Architecture Group—Enterprise Processor Division, Chandler, AZ, in 1997, where he has been engaged in the design of high-performance Itanium microprocessors with emphasis on circuit and physical design. He is presently working on the third-generation IA-64 processors.